Estimating the execution context for refining submission strategies on production grids

Diane Lingrand, Johan Montagnat, Tristan Glatard

Estimating the execution context for refining submission strategies on production grids.

Diane Lingrand, Johan Montagnat, and Tristan Glatard

University of Nice - Sophia Antipolis / CNRS - FRANCE
Diane.Lingrand@unice.fr, johan@i3s.unice.fr
http://www.i3s.unice.fr/~lingrand/

Abstract. In this paper, we study grid job submission latencies. The latency highly impacts performances on production grids, due to its high values and variations as well as the presence of outliers. It is particularly prejudicial for determining the status and expected duration of jobs.
In [1], a probabilistic model of the latency is presented that allows to estimate the best timeout value considering a given distribution of jobs latencies. This timeout value is then used in a job resubmission strategy. The purpose of this paper is to evaluate to what extent updating this model with relevant contextual parameters can help to refine the latency estimation. Experiments on the EGEE grid show that the choice of the resource broker or the computing site has a statistically significant influence on the jobs latency. We exploit this contextual information to propose a reliable job submission strategy.

1 Motivations

Production grids are characterized by permanent but non-stationary load and a large geographical extension. As a consequence, latency, measured as the time between the submission time of a computation job and the beginning of its execution, can be very high and experience large variations. As an example, on the EGEE grid[1] (Enabling Grid for E-sciencE), the average latency is in the order of 5 minutes with standard deviation also in the order of 5 minutes. This variability is known to highly impact application performances and thus has to be taken into account [2].

The main motivation for modeling the latency is to evaluate it precisely, hence giving a reliable estimation of the expected job completion time. On an unreliable grid infrastructure where a significant fraction of jobs is lost, this information is valuable to set up an efficient resubmission strategy minimizing the impact of faults. It can be exploited either at the workload management system level or at the user level. Too long running jobs are canceled and resubmitted before becoming too penalizing.

In [1], a probabilistic model of the latency is presented that allows to estimate the best timeout value considering a given distribution of jobs latencies. This timeout value is then used for job resubmission.

[1] EGEE, http://www.eu-egee.org/

In a previous work [3], we have shown that some parameters from the execution context have an influence on the cumulative density function of latency. In this paper, we quantify their influence on the timeout values and the expected execution time (including resubmissions). We aim at refining our model by taking into account most relevant contextual parameters in order to optimize our job resubmission strategy.

2 Related works

Several initiatives aim at modeling grid infrastructure Workload Management Systems (WMS). In [4], correlations between job execution properties (job size or number of processors requested, job run time and memory used) are studied on a multi-cluster supercomputer in order to build models of workloads, enabling comparative study on system design and scheduling strategies. In [5], authors make predictions of batch queue waiting time which improves the total execution time.

Taking into account contextual information has been reported to help in estimating single jobs and workflows execution time by rescheduling. Feitelson [6] has observed correlations between run time and job size, number of cluster and time of the day.

In [7], the influence of changes in transmission speed, in both executable code and data size, and in failure likelihood are analysed for a better estimation of end time of sub-workflows. This is used for re-scheduling jobs after fault or overrun.

Authors of [8] analyze job inter-arrival times, waiting times at the queues, execution times and data exchanged sizes. They made experiments on the EGEE grid on several VOs (Virtual Organizations) and studied the influence of the day of the week and the time of the day. Their conclusion on these influences is that there is an increase of the load at the end of the day and that it is difficult to extract a precise model of the behavior with respect of the day or the time.

To refine the grid monitoring, [9] presents a model of the influence between the grid components and their execution context (system and network levels), experimented on Grid'5000.

In this paper, we aim at refining our grid model with more local and dynamic parameters. Each job can be characterized by its execution context that depends on the grid status and may evolve during the job life-cycle. The context of a job depends both on parameters internal and external to the grid infrastructure. The internal context corresponds to parameters such as the computer(s) involved in the WMS of a specific job. It may not be completely known at the job submission time. The external context is related to parameters such as the day of the week that may be correlated to the grid workload.

3 Experimental platform

Our experiments are based on the EGEE production grid infrastructure. With 35000 CPUs dispatched world-wide in more than 240 computing centers, EGEE

represents an interesting case of study as it exhibits highly variable and quickly evolving load patterns that depend on the concurrent activity of thousands of potential users. The infrastructure is relatively homogeneous though as all computer hosting middleware services are state of the art PC-compatible computers running the same Operating System distribution (Scientific Linux v3) and hosted in computing centers with very high speed connections to the Internet.

For the following discussion, the main components of the batch-oriented EGEE grid infrastructure are introduced in figure 1.

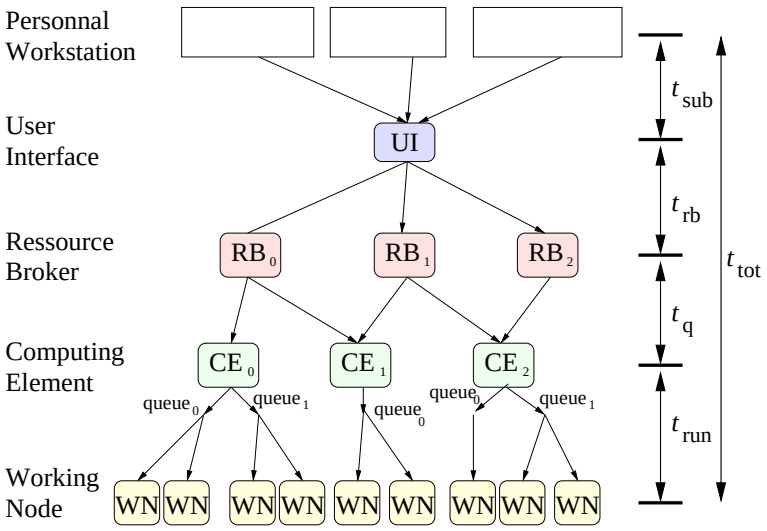

Fig. 1. EGEE job life cycle

When a user want to submit a job from her workstation, she connects to an EGEE client known as a User Interface (UI). A Resource Broker (RB) queues the user requests and dispatches them to the different computing centers available. The gateway to each computing center is one or more Computing Element (CE). A CE hosts a batch manager that will distribute the workload over the center Worker Nodes (WN), using different batch queues. Different queues are handling jobs with different wall clock time. However the policies for deciding of the number of queues and the maximal time assigned to each of them are site-specific.

During its life-cycle, a job is characterized by its evolving status. Received by the RB it is initially *waiting*, then *queued* at the CE and *running* on the WN. If everything went right, the job is then *completed*. Otherwise, it is *aborted*, *timed-out* or in an *error* status depending on the type of failure. As shown in figure 1, UIs can connect to different RBs, and RBs may be connected to overlapping sets of CEs.

4 Modelisation of the grid

Models of the grid latency enable the optimization of job submission parameters such as jobs granularity or the timeout value needed to make the WMS robust against system faults and outliers. Properly modeling a large scale infrastructure is a challenging problem given its heterogeneity and its dynamic behavior. In a previous work, we adopted a probabilistic approach [10] which proved to improve application performances while decreasing the load applied on the grid middleware by optimizing jobs granularities. Similar probabilistic models have been proposed to estimate timeouts in other complex systems [11, 12].

In [1], we show how the distribution of the grid latency impacts the choice of a timeout value for the jobs. We model the grid latency as a random variable R with probability density function (pdf) f_R and cumulative density function (cdf) F_R. The optimal timeout value can be obtained by minimizing the expectation of the job execution time J which can be expressed as a function of R, the timeout t_∞ and the proportion of outliers ρ:

$$E_J(t_\infty) = \frac{1}{F_R(t_\infty)} \int_0^{t_\infty} u f_R(u) du + \frac{t_\infty}{(1-\rho)F_R(t_\infty)} - t_\infty \tag{1}$$

Taking into account contextual information has recently been reported to help in estimating single jobs and workflows execution time by rescheduling [7]. We aim at refining our grid model with more local and dynamic parameters. Each job can be characterized by its execution context that depends on the grid status and may evolve during the job life-cycle. The context of a job depends both on parameters internal and external to the grid infrastructure. The internal context corresponds to parameters such as the computer(s) involved in the WMS of a specific job. It may not be completely known at the job submission time. The external context is related to parameters such as the day of the week and may have an impact on the load imposed to the grid.

Our final goal is to improve job execution performance on grids. This requires taking into account contextual information and its frequent update. In this paper, we are studying some parameters among the broad range of contextual information that could be envisaged and we discuss their relevance with regard to grid infrastructures.

5 Experimental data and experiences plan

To study the grid latency, measures were collected by submitting a very large number of probe jobs. These jobs, only consisting in the execution of an almost null duration `/bin/hostname` command, are only impacted by the grid latency. In the reminder we make the hypothesis that the users job execution time is known and that therefore only the grid latency varies significantly between different runs of the same computation task. To avoid variations of the system load, a constant number of probes was executing inside the system at any time of the data collection: a new probe was submitted each time another

one completed. For each probe job, we logged the job submission date, the UI used, the UI load at submission time, the RB used, the CE used and the jobs status duration (total duration t_{tot} and partial durations t_{sub}, t_{rb}, t_{q} and t_{run} as illustrated in figure 1). The probe jobs were assigned a fixed 10000 seconds timeout beyond which they were considered as outliers and canceled. This value is far greater than the average latency observed. In average in our measurements we observed a $\rho = 3\%$ ratio of outliers. We have observed that this ratio can increase significantly sometimes due to system faults though.

Three measure Data Sets are considered in this paper:

DS1. 5800 probe jobs acquired during 10 days in September 2006 over 3 RBs and 92 CEs.
DS2. 7233 probe jobs acquired during 1 week in April 2007 over 1 RB and 3 CEs.
DS3. 4173 probe jobs acquired during 1 week in May 2007 over 1 RB and 3 CEs.

These data sets were acquired randomly at very different times of the year to avoid unexpected correlation with external events. They cover all days of the week.

As an example, the cdf of the DS1 data set is plotted in figure 2. Its median is 363 seconds, its expectation is 570 seconds and its standard deviation is 886 seconds, which quantifies the highly variable behavior of the EGEE grid. The first part of this experimental distribution is close to a log-normal distribution and its tail can be modeled by a Pareto distribution [1]. This heavy tailed distribution shows that the EGEE grid exhibits non-negligible probabilities for long latencies.

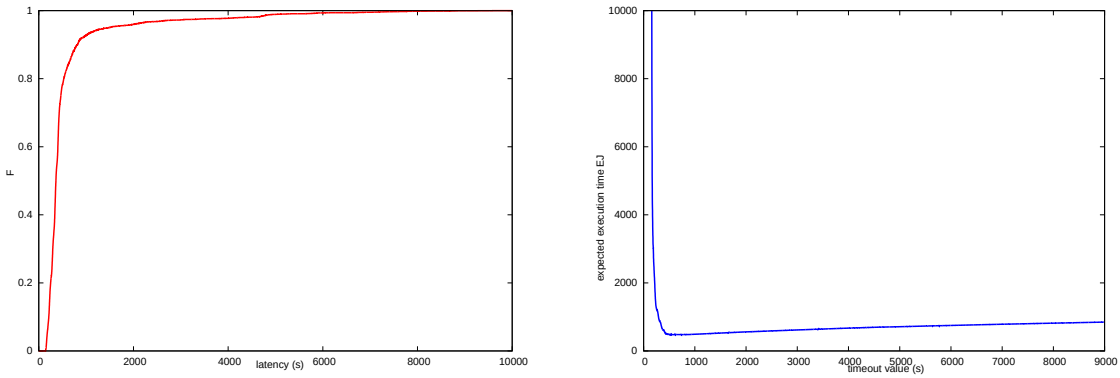

Fig. 2. Left: cumulative density function of the whole experimental data. Right: expectation of the job execution time (in seconds), including resubmission, with respect to the timeout value t_∞ (in seconds). The minimum of this curve gives the best timeout value. Here, the best timeout is $t_\infty = 556s$ giving $E_J = 479s$.

In the remainder of the paper these measurements are exploited to quantify the jobs latency and to evaluate the impact of various internal and external context parameters. A context-dependent optimal timeout value is thus computed. It is the basis of an optimal resubmission strategy that aims at minimizing the expected execution time of jobs submitted to the grid infrastructure. In particular, we consider in the following sections the impact of the target RB and the target CE which are expected to have an influence on the latency due to variable computing sites performance and variable load conditions. In addition, the correlation between time of the day and latency is studied since external parameters such as working and week-end days are expected to be correlated to different system loads as well.

6 Influence of the Resource Broker (RB).

In this experiment, three different Resource Brokers were considered: a french one (`grid09.lal.in2p3.fr`), a spanish one (`egeerb.ifca.org.es`) and a russian one (`lcg16.sinp.msu.ru`). Their cdfs are shown in figure 3. The optimal timeout values computed and the resulting expected execution time are reported below. The table displays:

- the optimal timeout value estimated and the difference between this value and the global reference value obtained using all measurements without distinction;
- the minimal expected execution time;
- the expected execution time if the timeout is set to the global reference value and the difference with the optimum.

	all RBs	RB fr	RB es	RB ru
optimal t_∞	$t_{\infty\text{ref}} = 556$ s	729 s	546 s	506 s
Δt_∞	0%	31%	2%	9%
best E_J	479.125 s	483.7 s	445.2 s	476.2 s
$E_J(t_{\infty\text{ref}})$	479.125 s	488.8 s	445.9 s	477.9 s
ΔE_J	0%	1%	0.2%	0.4%

The optimal timeout values obtained differ significantly and the most distinct is the one associated to the Spanish RB (variation of 31%). However, the expected execution time varies by a much smaller amount (1% maximum). This is related to the fact that in this case (relatively low outliers ratio and rather homogeneous infrastructure), slightly overestimating the timeout has little impact on the execution time. It should be noted that an underestimation is impacting the execution time much more though as can be seen in figure 4.

To simulate a more variable infrastructure, we applied the model considering a variable level of outliers between the different RBs (ρ = 20%, 3% and 0% respectively). These errors are realistic as error conditions regularly lead to outliers ratios as high as 20%. The results are summarized in the following table:

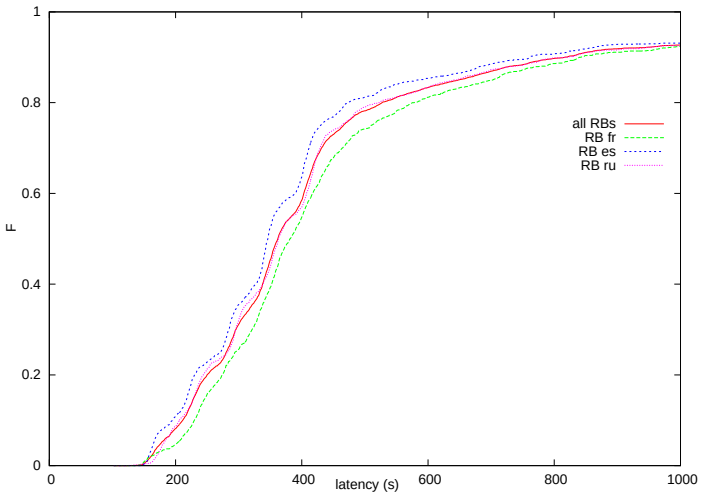

Fig. 3. Cumulative density function for the different Resource Brokers.

	all RBs	RB fr	RB es	RB ru
ρ	7.7%	20%	3%	0%
optimal t_∞	$t_{\infty \text{ref}} = 868$ s	551 s	546 s	865 s
best E_J	452.3 s	639.8 s	445.2 s	451.7 s
$E_J(t_{\infty \text{ref}})$	452.3 s	691.7 s	456.2 s	451.7 s
ΔE_J	0%	8%	2.5%	0%

In this case, the model consistently reports growing execution time disruptions with the increase of the number of outliers. The resubmission strategy still rather efficiently cope with the errors as the execution time variation does not exceed 8%. Taking into account the submission RB can help in adapting the optimal timeout choice. The more variable the infrastructure, the more valuable the optimization.

7 Influence of the Computing center

In a computing center, the batch submission system is usually configured with several queues. The influence of the Computing Element (CE) and the associated queues, later abbreviated as CE-queue, is considered in this section. The same methodology than with RBs in section 6 could be envisaged but a significant difference is that the number of CE-queues is much larger than the number of RBs: in our experiment, we had 92 CEs and queues and 3 RBs. It might thus

be relevant to group similar CE-queues to obtain fewer classes. As can be seen in figure 4 many of the 92 CE-queues have similar cdfs while others are more singular.

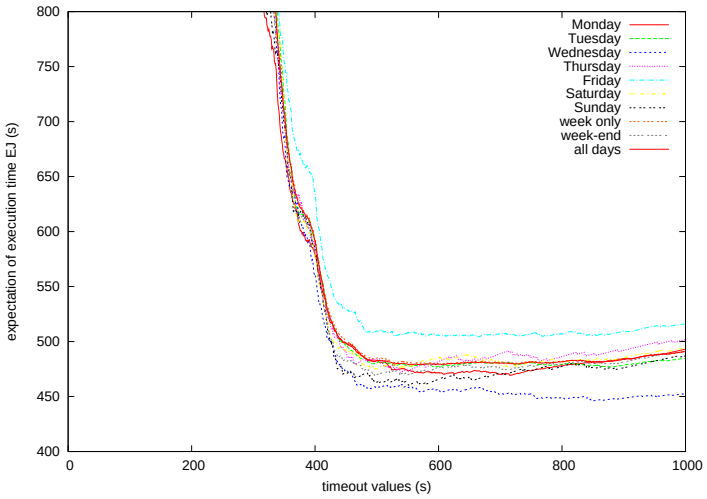

Fig. 4. Expectation of job execution time with respect to the timeout value (t_∞) for the different CE and queues.

The idea we promote here is to group CEs and queues that have similar properties into different classes.

7.1 Classification of the CE and queues

Different aggregations of CE-queues were tested based on their cdf using the k-means classification algorithm with $k = 2$ to 10 classes. For each CE-queue entity, the cdf has been computed. From this cdf the optimal timeout value is computed , by minimizing equation 1. Figures 5 and 6 show the repartition of the timeout values in the classes. The depth of each box is proportional to the number of CE-queues in the class.

In order to measure if the classes are statistically discriminant, we have tested the hypothesis H_0 (all set have equal mean and equal variance) using ANOVA (ANalysis Of VAriance). The results are reported in the following table (*** means rejection of hypothesis H_0 with high confidence):

nb. of classes	Df	Sum Sq	Mean Sq	F value	Pr(>F)	H_0 rejection
2	1	1304048	1304048	13.895	0.0003643	***
3	2	1777728	888864	9.9968	0.0001381	***
4	3	3078172	1026057	14.061	2.221e-07	***
5	4	3061326	765331	10.318	1.035e-06	***
6	5	3133050	626610	8.4443	2.324e-06	***
7	6	4083366	680561	10.941	1.165e-08	***
8	7	4161760	594537	9.5927	2.29e-08	***
9	8	4464774	558097	9.5265	7.536e-09	***
10	9	4450327	494481	8.2929	2.73e-08	***

The result of the ANOVA test shows that these means differ significantly with a probability lower than 0.1% in all cases. The best result is obtained for 9 classes but the gain is not so high. Note that the ANOVA test only show that the hypothesis H_0 is rejected: this does not necessary imply that all classes differ from each other.

In the case of 2 classes, these classes are statistically discriminant. But for all other cases, further tests must be done in order to determine how many classes are independent.

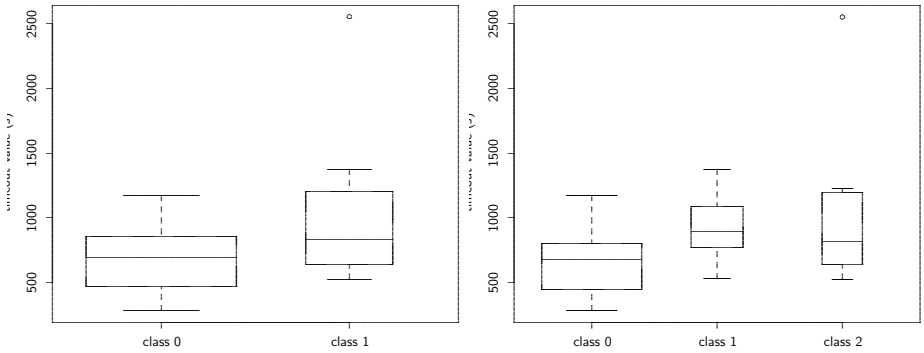

Fig. 5. Timeout values repartition after k-mean classification into 2 classes (on the left) and 3 classes (on the right) of CEs and queues.

7.2 Refining the ANOVA analysis

Let us look, for example, at the case of classification into 3 different classes (classes 0, 1 and 2). Using ANOVA, if we test classes 1 and 2, we observe that they do not differ significantly: $F = 0.2334$ ($p < 0.6338$). Building a new class, class 1+2, from classes 1 and 2, we now test class 0 against class 1+2 and obtain that they differ significantly: $F = 19.651$ ($p < 3.003e - 05$). We observe that

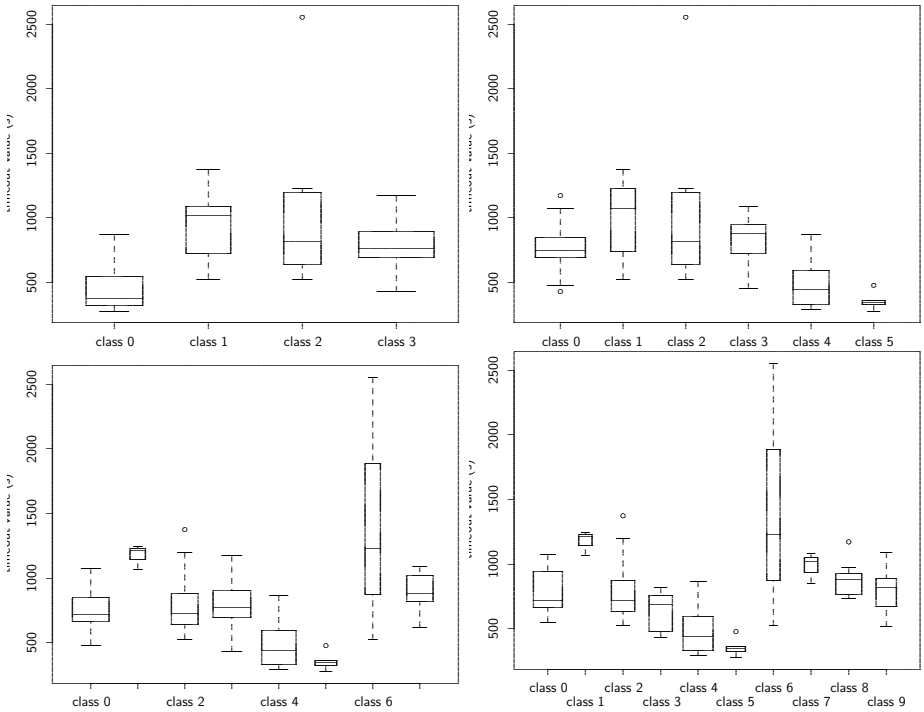

Fig. 6. Timeout values repartition after k-mean classification into 4, 6, 8 and 10 classes of CE-queues.

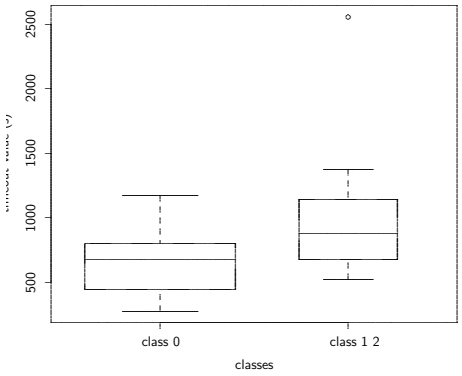

Fig. 7. k-means classification into 3 classes after grouping classes 1 and 2.

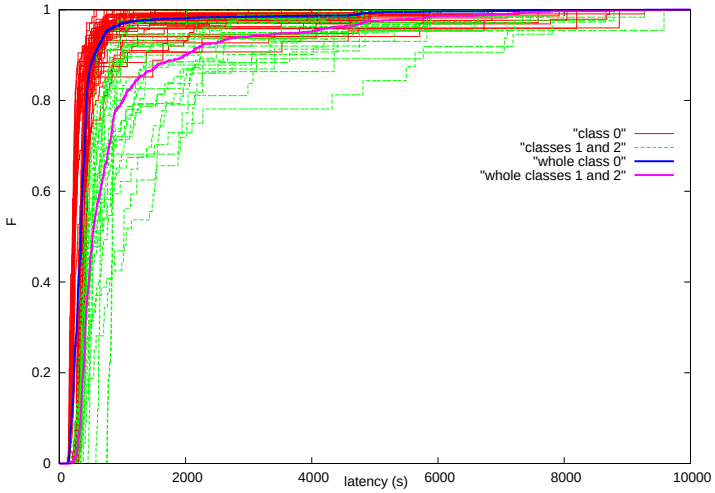

Fig. 8. Cumulative density functions of latency with respect to time (in seconds). This figure is obtained from the k-means classification into 3 classes. We grouped the last 2 classes into a single one so that we have 2 classes: the initial class 0 (in red) and new class 3 (in green) resulting of the merging of classes 1 and 2. Each curve corresponds to the cdf of one CE-queue. The curves in blue and magenta correspond to the global cdf for class 0 (in blue) and class 3 (in magenta).

grouping two classes after the classification $k = 3$ gives a similar although slightly better result than the classification $k = 2$.

The optimal timeout for the 2 populations (classes 0 and 3) are $t_{\infty 0} = 779s$ and $t_{\infty 3} = 881s$ respectively (see figure 8).

Figure 9 shows the errors computed between the best E_J and $E_J(t_\infty)$, where t_∞ if computed from the whole class.

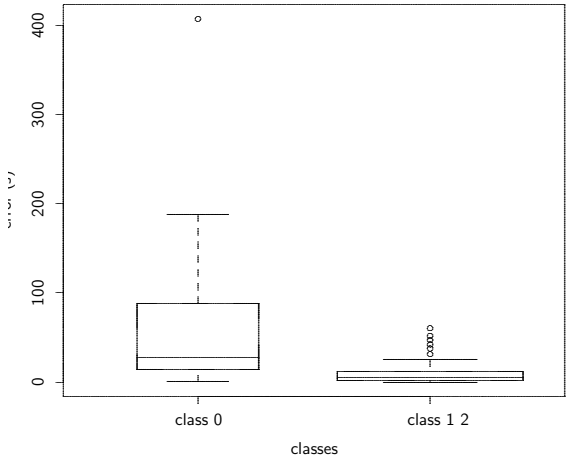

Fig. 9. Errors measured between best E_J and $E_J(t_\infty)$, where t_∞ if computed from the whole class.

8 Influence of the days of the week combined with RBs

The day of the week potentially can influence the optimal timeout value since the activity is likely to be maximal during the week days (thus increasing the system load) and minimal during week-ends. Conversely, system maintenance is expected to be better during week days than during week-ends.

It this experiment, the day of the week information is correlated with the Resource Broker to evaluate the mutual influence of both parameters. All measurements from DS1, DS2 and DS3 were gathered to avoid a bias due to a particular activity at a given week of the year. The optimal timeout values are reported in the following table for each day of the week, the week only and the week-end considering either all measurements or per-RB measures. The minimum optimal timeout value is 436 seconds while the maximum is 1105 seconds. The difference is about 11 minutes. Taking into account both RB and day values can thus drastically improve the total job execution time.

/seconds nb. of data	Mon 1223	Tue 1782	Wed 547	Thu 299	Fri 1120	Sat 487	Sun 382	week only 4971	week-end 869	all days 5840
all RBs	716	880	1075	538	855	715	551	868	551	868
RB fr	715	879	1066	762	1105	881	579	864	886	866
RB es	566	880	930	538	479	726	521	562	703	546
RB ru	699	879	1075	640	849	495	436	864	495	865

Figure 10 shows the different cdfs with respect to the day of the week. We observe that the curves corresponding to Saturday and Sunday are slightly below the other curves, meaning that the latency is higher on week-end. Figure 11 shows the corresponding optimal timeout values computed by minimizing equation 1.

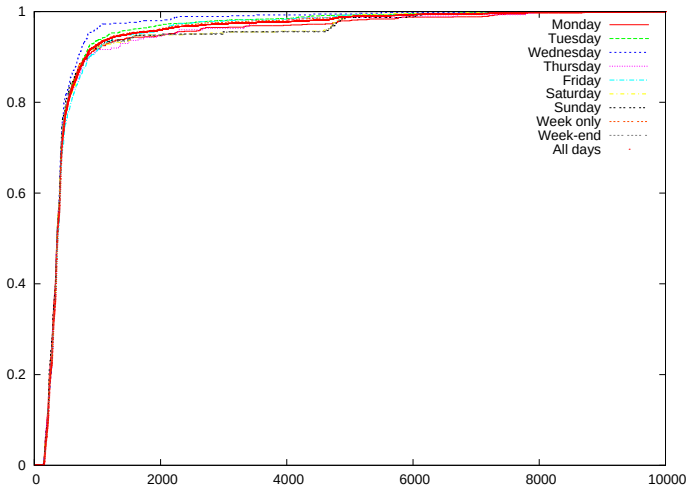

Fig. 10. Cumulative density function with respect to the timeout value for different days of the week.

The timeout for all these weeks and for all days of the week are plotted in figure 12. It can be noted that surprisingly, the optimal timeout values are much lower than reported in the previous experiments: in the order of 30 seconds. Other independent studies confirmed this phenomenon: about 25% of the probe jobs in DS2 and DS3 finish their execution in less than 50 seconds while the remaining 75% take much longer to complete (in the order of 300s). This behavior could not be statistically correlated to a specific computing site, queue, nor resource broker. It might depend on other non-trivial factors such as the workload management

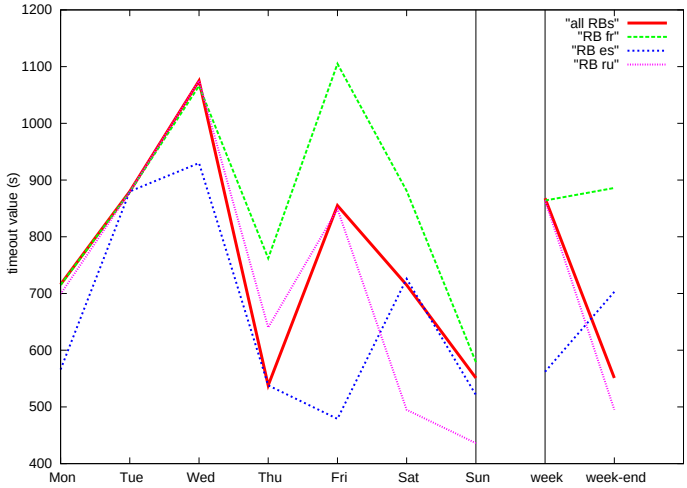

Fig. 11. Optimal timeout values for different RB and with respect to the day of the week.

system strategy or some sites scheduling policy. Figure 13 shows the variations of the expectation of the job execution time with respect to the timeout.

Using the low timeout is probably a safe and efficient strategy as long as the infrastructure is not overloaded (*i.e.* as long as few users do that): it causes massive resubmission of jobs after a short time knowing that a significant fraction of jobs finishes early. However, there is a risk of overload due to this aggressive strategy. To avoid that, we also computed a second optimal value that corresponds to the optimal among values greater than 200 seconds with values closer to the results reported on DS1 (figure 14).

However, some behaviors are common to the different sets of data: increase of timeout values in the beginning of the week, short decrease and increase in the middle of the week and finally, decrease during the week-end (from Saturday to Sunday).

This experiment show the high variability of the grid infrastructure load conditions and the need to acquire data for long periods of time (several months) in order to observe the variable load patterns. Further work will be invested in estimating the required frequencies of updates of the statistics collection over the grid.

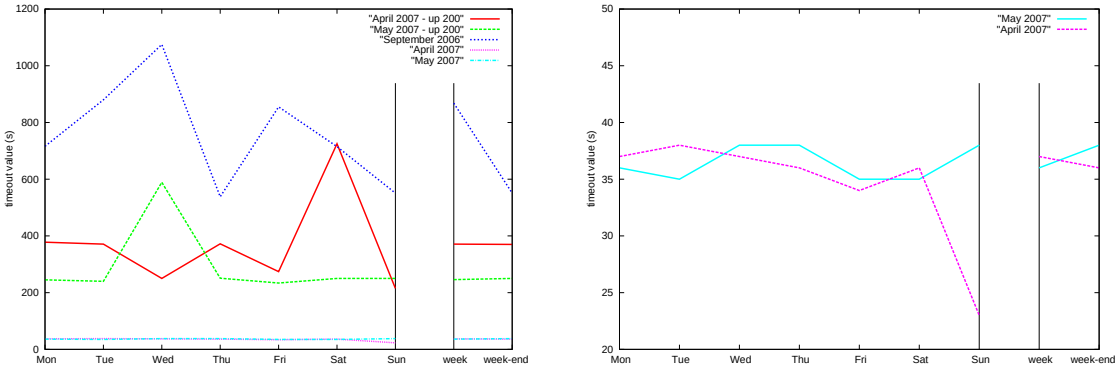

Fig. 12. Optimal timeout values with respect of the day of the week using three different weeks. On the right, we detail the best timeout values.

9 Conclusions

In this paper, we have shown that day of the week, Resource Brokers and CE-queues have an influence on the expected job execution time. Moreover, we have shown that we can group CE-queues into classes that are statistically different, reducing the number of data to be analyzed. The statistics collected can be used to estimate the jobs execution time and optimize the job submission procedure.

The methodology used could be applied to other grids by replacing CEs and RBs by the equivalent workload management services. In the DIET middleware [13] for instance, it could correspond to Master Agents (MA) and Local Agents (LA).

However, the study on the influence of the day of the week need to be continued by collecting more data (acquisition of logs during several months) to determine (i) if there are global trends observable during several weeks and (ii) how frequently the experimental models need to be updated.

Acknowledgments

This work is partially funded by the French national research agency (ANR), NeuroLog project[2] under contract number ANR-06-TLOG-024. This project is in the scope of scientific topics of the STIC-ASIA OnCoMedia project[3]. We are grateful to the EGEE European project for providing the grid infrastructure and user assistance.

[2] Neurolog: http://neurolog.polytech.unice.fr
[3] OnCoMedia: http://www.onco-media.com

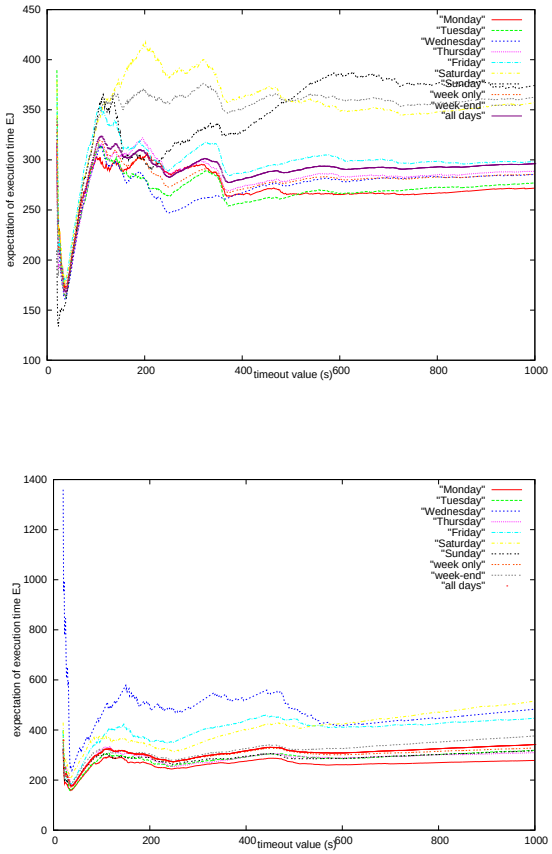

Fig. 13. Expectation of job execution time with respect to the timeout value (t_∞) for the different days of the week. Left: SD2. Right: SD3.

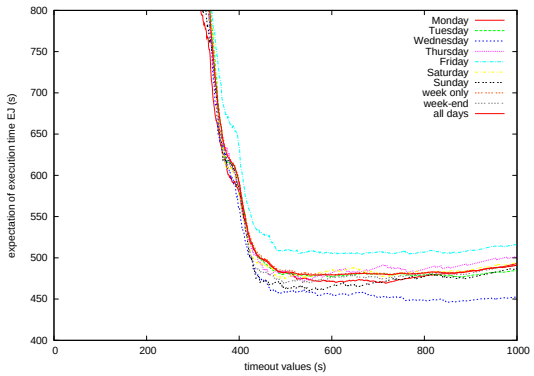

Fig. 14. Expectation of job execution time with respect to the timeout value (t_∞) for the different days of the week. Data from September 2006.

References

1. Glatard, T., Montagnat, J., Pennec, X.: Optimizing jobs timeouts on clusters and production grids. In: International Symposium on Cluster Computing and the Grid (CCGrid), Rio de Janeiro, IEEE (2007) 100–107
2. Schopf, J., Berman, F.: Stochastic Scheduling. In: Supercomputing (SC'99), Portland, USA (1999)
3. Glatard, T., Lingrand, D., Montagnat, J., Riveill, M.: Impact of the execution context on Grid job performances. In: International Workshop on Context-Awareness and Mobility in Grid Computing (WCAMG07), Rio de Janeiro, IEEE (2007) 713–718
4. Li, H., Groep, D., Walters, L.: Workload Characteristics of a Multi-cluster Supercomputer. In: Job Scheduling Strategies for Parallel Processing, Springer Verlag (2004) 176–193
5. Nurmi, D., Mandal, A., Brevik, J., Koelbel, C., Wolski, R., Kennedy, K.: Evaluation of a Workflow Scheduler Using Integrated Performance Modelling and Batch Queue Wait Time Prediction. In: Conference on High Performance Networking and Computing, Tampa, Florida, ACM Press (2006)
6. Feitelson, D. In: Workload modeling for performance evaluation. Springer-Verlag - LNCS vol 2459 (2002) 114–141
7. Nichols, J., Demirkan, H., Goul, M.: Autonomic Workflow Execution in the Grid. IEEE Transactions on Systems, Man, and Cybernetics **36** (2006)
8. Oikonomakos, M., Christodoulopoulos, K., Varvarigos, E.: Profiling Computation Jobs in Grid Systems. In: IEEE International Symposium on Cluster Computing and the Grid (CCGrid07), Rio de Janeiro, Brasil (2007) 197–204
9. Ravelomanana, S., Bianchi, S.C.S., Joumaa, C., Sibilla, M.: A Contextual GRID Monitoring by a Model Driven Approach. In: Proceedings of the Advanced International Conference on Telecommuni cations and International Conference on

Internet and Web Applications and Servic (AICT/ICIW), Gosier, Guadeloupe, IEEE (2006) 37–43
10. Glatard, T., Montagnat, J., Pennec, X.: Probabilistic and dynamic optimization of job partitioning on a grid infrastructure. In: 14th euromicro conference on Parallel, Distributed and network-based Processing (PDP06), Montbéliard-Sochaux, France (2006) 231–238
11. van Moorsel, A., Wolter, K.: Analysis of Restart Mechanisms in Software Systems. IEEE Transactions on Software Engineering (TSE) **32** (2006) 547–558
12. Libman, L., Orda, A.: Optimal Retrial and Timeout Strategies for Accessing Network Resources. IEEE/ACM Transactions on Networking (TN) **10** (2002) 551–564
13. Caron, E., Desprez, F.: DIET: A Scalable Toolbox to Build Network Enabled Servers on the Grid. International Journal of High Performance Computing Applications (2005)